УДК: 519.681

А.Е. Дорошенко, В.Г. Акуловский

Алгоритмическое описание взаимодействия алгоритмов с внешними устройствами

Алгебра алгоритмов с данными дополнена средствами описания операций ввода-вывода в результате модификации абстрактной модели ЭВМ Глушкова. Возможность описания операций ввода-вывода актуальна для широкого класса систем, интенсивно взаимодействующих с внешними устройствами. Перспективность полученных возможностей показана на примере простейшей системы управления.

The algebra of the algorithms with data is supplemented by the descriptions of input/output operations resulting in the modification of the Glushkov's abstract model computer. The ability to describe the input/output operations is relevant to a wide range of systems, which are intensely interacting with external devices. The prospect of capabilities is demonstrated at the example of a simple control system.

Алгебру алгоритмів з даними доповнено засобами опису операцій вводу-виводу в результаті модифікації абстрактної моделі ЕОМ Глушкова. Можливість опису операцій вводу-виводу актуальна для широкого класу систем, що інтенсивно взаємодіють із зовнішніми пристроями. Перспективність отриманих можливостей показано на прикладі простої системи управління.

Введение. Ключевую роль в процессе разработки алгоритмов программных систем [1–3] играют данные. В результате модификации известной [4] модели ЭВМ Глушкова предложена алгебра алгоритмов с данными (САА\Д) [5, 6]. САА\Д – это система алгоритмических алгебр $<U, L, \Omega>$, где U – множество Д-операторов, L – множество логических условий, Ω – сигнатура операций, состоящая из логических операций Ω_1, принимающих значения на множестве L, и операций Ω_2, принимающих значения на множестве операторов U.

Постановка задачи

Принципиальные отличия САА\Д от алгебры Глушкова состоят в следующем.

Модификация модели ЭВМ заключается в том, что операционный автомат, входящий в указанную модель, оснащен памятью – носителем данных.

Данными D, хранящимися в памяти, называется упорядоченная пара $<\Delta,З>$, где Δ – носитель данных (фрагмент памяти), З – кортеж значений, хранимый этим носителем данных в текущий момент времени.

Введено понятие – состояние вычислительного процесса.

Определение 1. Состояние вычислительного процесса на любом i-м шаге выполнения $D_i^T=D_i^P\cup D_i^d\cup\alpha_i^{k+1}$ определяется статической составляющей (текущим состоянием памяти D_i^P – статические данные) и динамической составляющей D_i^d и α_i^{k+1}, где D_i^d – текущее состояние множества динамических данных, α_i^{k+1} – состояние логического условия. Статическая составляющая имеет место на каждом шаге вычислительного процесса, т.е. $D_i^P\neq\varnothing$ при всех возможных i. Динамическая составляющая определена только на некоторых его шагах, т.е. $\exists p$, где $D_p^d\neq\varnothing$ в состоянии D_p^T и $\exists m$, где $\{\alpha_m^{k+1}\}\neq\varnothing$ в состоянии D_m^T. Эта составляющая существует только на данном шаге, т.е. для любого D_j^T, где $j\neq p$ и $j\neq$m, выполняется $D_p^d=\varnothing$ и $\{\alpha_m^{k+1}\}=\varnothing$. При этом, если $D_p^d\neq\varnothing$, то $\{\alpha_p^{k+1}\}=\varnothing$, а если $\{\alpha_m^{k+1}\}\neq\varnothing$, то $D_m^d=\varnothing$.

Из определения следует, что возможны следующие состояния вычислительного процесса: $D_i^T=D_i^P$ ($D_i^d=\varnothing$, $\{\alpha_i^{k+1}\}=\varnothing$), $D_i^T=D_i^P\cup D_i^d$ ($\{\alpha_i^{k+1}\}=\varnothing$), $D_i^T=D_i^P\cup\{\alpha_i^{k+1}\}$ ($D_i^d=\varnothing$).

Д-операторы изменяют состояние вычислительного процесса, переводя его из исходного для него состояния D_i^T в состояние D_{i+1}^T такое,

что $D_{i+1}^T \neq D_i^T$. Они в общем случае записываются в виде $(\mathbf{D})X(\mathbf{D}')$, где $\mathbf{D}$ и $\mathbf{D}'$ – данные, специфицированные на его входе и выходе, чем и обусловлено название операторов. На входе и выходе Д-операторов, в соответствии с определением 1, специфицируются статические, динамические данные и логические условия.

Статические данные $D \subseteq \mathbf{D}$ и $D' \subseteq \mathbf{D}'$ – это кортежи значений, хранимые в памяти $\mathbf{D} \subseteq D^p$, $\mathbf{D}' \subseteq D^p$ и влияющие на состояние вычислительного процесса $D_i^P \subseteq D_i^T$ на каждом его шаге.

Динамические данные $D_i^d \subseteq \mathbf{D}' \subseteq D_i^T$ – это кортеж значений, который не хранится в памяти, а определен (существует) только на текущем шаге вычислительного процесса.

Логические условия $\alpha_i^{k+1} \in \mathbf{D}' \subseteq D_i^T$ в общем случае $k+1$-значные $\alpha^{k+1} \in \{0,1,...,k-1,\mu\}$, где μ – неопределенное значение логического условия. В частных случаях логические условия могут быть произвольной значности, например, для трехзначных логических условий $\alpha^3 \in E_3 = \{0,1,\mu\}$. Логические условия определены $\{\alpha_i^{k+1}\} \subseteq D_i^T$ (существуют) только на текущем шаге вычислительного процесса.

Д-операторы образуют следующий базовый набор:

- $(D)O(D')$ переводит вычислительный процесс в состояние $D_{i+1}^T = D_{i+1}^P$ такое, что $D_{i+1}^P \neq D_i^P$, т.е. изменяет состояние памяти;
- $(D)O(D^d)$ переводит вычислительный процесс в состояние $D_{i+1}^T = D_{i+1}^P \cup D_{i+1}^d$ такое, что $D_{i+1}^d \neq \varnothing$, $D_{i+1}^P = D_i^P$, т.е. определяет (продуцирует) динамические данные;
- $(D^\pi)\mathrm{P}(\alpha^{k+1})$ представляет собой в общем случае n-местную логическую функцию, называемую предикатом. Предикат переводит в состояние $D_{i+1}^T = D_{i+1}^P \cup \{\alpha_{i+1}^{k+1}\}$ такое, что $\{\alpha_{i+1}^{k+1}\} \neq \varnothing$, $D_{i+1}^P = D_i^P$, т.е. определяет (продуцирует) $k+1$-значное (в общем случае) логическое условие.

Исходя из определения 1, введено понятие тождественного Д-оператора.

Определение 2. Д-оператор $(D)X(D')$, в результате исполнения которого получено соотношение $D_i^T = D_{i+1}^T$, будем называть тождественным и обозначать Z.

Из определения 2 и определения Д-операторов следует

$$(D)X(\varnothing) = Z, \ (\varnothing)X(\varnothing) = Z. \qquad (1)$$

Все возможные Д-операторы образуют множество U, на котором определены операции САА\Д.

Операция композиция (обозначается «*») Д-операторов $(\mathbf{D}_1)X_1(\mathbf{D}_1')*(\mathbf{D}_2)X_2(\mathbf{D}_2')$ означает последовательное их выполнение.

Операция p_k-дизъюнкции

$$[(D^\pi)P(\alpha^{k+1})]((D_1)O_1(D_1') \vee (D_2)O_2(D_2') \vee ...$$
$$... \vee (D_{k-1})O_{k-1}(D_{k-1}') \vee (D_k)O_k(D_k')) =$$
$$= \begin{cases} (D_1)O_1(D_1'), \text{ если } \alpha^{k+1} = k-1; \\ (D_2)O_2(D_2'), \text{ если } \alpha^{k+1} = k-2; \\ \dots\dots\dots\dots \\ (D_k)O_k(D_k'), \text{ если } \alpha^{k+1} = 0; \\ (D_{k+1})O_{k+1}(D_{k+1}'), \text{ если } \alpha^{k+1} = \mu. \end{cases}$$

Результат выполнения этой операции – один из $k+1$ возможных Д-операторов, который выбирается в соответствии со значением логического условия α^{k+1}.

В частных случаях, это операция p_2-дизъюнкции

$$[(D^\pi)P(\alpha^2)]((D_1)O_1(D_1') \vee (D_2)O_2(D_2')) =$$
$$= \begin{cases} (D_1)O_1(D_1'), & \text{если } \alpha^2 = 1; \\ (D_2)O_2(D_2'), & \text{если } \alpha^2 \neq 1, \end{cases}$$

где, в соответствии со значением логического условия α^2, выбирается один из двух возможных Д-операторов и другие производные операции p-дизъюнкции.

Операция p-итерации

$[(D^\pi)P(\alpha^2)]\{(D)O(D')\}$ осуществляет циклическое выполнение Д-оператора $(D)\mathrm{O}(D')$ (тела цикла) при $\alpha^2 = 1$ и завершается в противном случае.

Д-оператор в рамках САА\Д может быть представлен в следующей форме.

Определение 3. Представление любого Д-оператора из U через образующие элементы системы $<U, L, \Omega>$ называется регулярной схемой этого Д-оператора (РСД).

Поскольку любая операция САА\Д может рассматриваться как Д-оператор, определим еще одну форму представления Д-операторов.

Определение 4. Композиционной схемой (КС) Д-оператора из U называется представление этого Д-оператора в виде композиции других Д-операторов, с помощью которых, в частности, представлены операции сигнатуры САА\Д из множества Ω_2.

На некоторых этапах разработки и/или при описании некоторых классов алгоритмов могут совместно использоваться обе формы записи Д-операторов, такую форму записи назовем композиционно-регулярной схемой (КРС).

В приведенном алгебраическом аппарате средства, ориентированные на описание взаимодействия алгоритма с внешними устройствами (ВУ), отсутствуют. При этом известно, что большинство программных систем интенсивно с ними взаимодействует и, очевидно, что такое взаимодействие существенным образом влияет на свойства алгоритма. Чтобы восполнить указанный пробел расширим возможности САА\Д.

Формализация операций ввода-вывода

Для обеспечения возможности описания алгоритмов, взаимодействующих с ВУ, выполним еще одну модификацию модели ЭВМ и покажем ее на рисунке.

Будем полагать, что ВУ являются носителями данных, организованных следующим образом. Элементарным носителем данных есть бит (разряд). Некоторая совокупность элементарных носителей данных образуют простые данные, в общем случае различные. Некоторая совокупность простых носителей данных образуют составные носители данных. Эти носители данных являются их источниками и/или приемниками. Подчеркнем, что не все устройства функционируют как источники и приемники данных, некоторые из них выполняют одну из этих функций.

Будем различать три типа ВУ, т.е. ВУ = {ВП, УВВ, УСО}, где ВП – множество устройств внешней памяти $\text{ВП} = \{\text{ВП}_1, \text{ВП}_2, \ldots, \text{ВП}_m\}$, УВВ – множество устройств ввода-вывода $\text{УВВ} = \{\text{УВВ}_1, \text{УВВ}_2, ..., \text{УВВ}_n\}$, УСО – множество устройств связи с объектом управления $\text{УСО} = \{\text{УСО}_1, \text{УСО}_1, ..., \text{УСО}_k\}$.

Каждое устройство ВП будем рассматривать как множество файлов $\text{ВП}_i = \{F_1^{\text{ВП}_i}, F_2^{\text{ВП}_i}, ..., F_r^{\text{ВП}_i}\}$, а каждый файл как носитель данных, образованный простыми или составными носителями данных $F_j^{\text{ВП}_i} = \{\Delta_1^{F_j^{\text{ВП}_i}}, \Delta_2^{F_j^{\text{ВП}_i}}, ..., \Delta_n^{F_j^{\text{ВП}_i}}\}$.

Каждое УВВ и УСО будем рассматривать как носитель данных, образованный простыми или составными носителями данных $\text{УВВ}_p = \{\Delta_1^{\text{УВВ}_p}, ..., \Delta_m^{\text{УВВ}_p}\}$ и $\text{УСО}_s = \{\Delta_1^{\text{УСО}_s}, ..., \Delta_r^{\text{УСО}_s}\}$.

Данными D, как и в случае с памятью, назовем упорядоченную пару $<\Delta,З>$, где Δ – носитель данных, З – кортеж значений, хранимый этим носителем данных в текущий момент времени.

Поскольку ВУ представляют собой некоторую иерархию, то для записи вводимых и выводимых данных в зависимости от степени детализации алгоритма будем использовать следующие обозначения: $D^{R\_BУ}$, $D^{W\_BУ}$, когда речь идет о всех внешних устройствах; $D^{R\_\text{ВП}}$, $D^{R\_\text{УВВ}}$, $D^{R\_\text{УСО}}$, $D^{W\_\text{ВП}}$, $D^{W\_\text{УВВ}}$, $D^{W\_\text{УСО}}$, когда речь идет о группах устройств; $D^{R\_\text{ВП}_i}$, $D^{R\_\text{УВВ}_i}$, $D^{R\_\text{УСО}_i}$, $D^{W\_\text{ВП}_i}$, $D^{W\_\text{УВВ}_i}$, $D^{W\_\text{УСО}_i}$, $F_j^{R\_\text{ВП}_i}$, $F_j^{W\_\text{ВП}_i}$, когда речь идет об отдельном i-м устройстве или файле на устройстве внешней памяти в последнем случае; $D_k^{R\_\text{УВВ}_i}$, $D_k^{R\_\text{УСО}_i}$, $D_k^{W\_\text{УВВ}_i}$, $D_k^{W\_\text{УСО}_i}$, $D_k^{R\_F_j^{\text{ВП}_i}}$, $D_k^{W\_F_j^{\text{ВП}_i}}$, когда речь идет о конкретном k-м множестве данных, вводимых или выводимых на i-е ВУ, j-й файл i-го устройства ВП в последнем случае.

В случаях, когда речь будет идти о вводимых или выводимых данных вне зависимости

от типа ВУ и уровня иерархии, такие данные в дальнейшем будем обозначать D^R и D^W соответственно.

Учитывая выполненную модификацию модели ЭВМ, доопределим понятие состояния вычислительного процесса.

Определение 5. Статическая составляющая состояния вычислительного процесса, определенная на каждом его шаге, включает состояние внешних устройств $D^{\text{ВУ}}$ и, таким образом, на любом i-м шаге вычислительный процесс находится в одном из трех возможных состояний: $D_i^T=D_i^P\cup D_i^{\text{ВУ}}$, $D_i^T=D_i^P\cup D_i^{\text{ВУ}}\cup D_i^d$, $D_i^T=D_i^P\cup \cup D_i^{\text{ВУ}}\cup\{\alpha_i^{k+1}\}$. В остальном данное определение полностью совпадает с определением 1.

Исходя из определения 5, в базовый набор введем дополнительные Д-операторы, которые переводят вычислительный процесс из любого исходного состояния D_i^T в состояние $D_{i+1}^T\neq D_i^T$, где $\{\alpha_{i+1}^{k+1}\}=\varnothing$, следующим образом:

- $(D)\ O\ (D^W)$ выводит данные из памяти на ВУ и переводит вычислительный процесс в такое состояние, что $D_{i+1}^{\text{ВУ}}\neq D_i^{\text{ВУ}}$, $D_{i+1}^P=D_i^P$;
- $(D^d)\ O\ (D^W)$ выводит динамические данные на ВУ и переводит вычислительный процесс в такое состояние, что $D_{i+1}^{\text{ВУ}}\neq D_i^{\text{ВУ}}$, $D_{i+1}^P=D_i^P$, $D_{i+1}^d=\varnothing$;
- $(D^R)\ O\ (D')$ вводит данные из ВУ и переводит вычислительный процесс в состояние такое, что $D_{i+1}^P\neq D_i^P$, $D_{i+1}^{\text{ВУ}}=D_i^{\text{ВУ}}$, $D_{i+1}^d=\varnothing$;
- $(D^R)\ O\ (D^d)$ вводит данные с ВУ, определяет состояние динамической памяти и переводит вычислительный процесс в такое состояние, что $D_{i+1}^d\neq\varnothing$, $D_{i+1}^{\text{ВУ}}=D_i^{\text{ВУ}}$, $D_{i+1}^p=D_i^p$;
- $(\varnothing)S(D^W)$ – выполняет служебные (вспомогательные) функции, например, включает, выключает, осуществляет настройки ВУ. Этот Д-оператор переводит вычислительный процесс в такое состояние, что $D_{i+1}^{\text{ВУ}}\neq D_i^{\text{ВУ}}$, $D_{i+1}^p=D_i^p$, $D_{i+1}^d=\varnothing$.

Введенные Д-операторы, выполняющие операции ввода-вывода, расширяют функциональность, описывающих их схем.

Композиционные схемы алгоритмов

Из определений 3 и 4 следует, что могут быть построены производные Д-операторы. Например, $(D_1)Q_1(D_1')*(D_2)Q_2(D^d)=(D_3)Q_3(D^d)$, $(D)Q(D^d)*$ $*(D^\pi)P(\alpha^{k+1})=(D_1)Q_1(\alpha^{k+1})$ и т.д. В производных Д-операторах часть данных и функциональности инкапсулируется, т.е. они представляют собой более общие случаи Д-операторов. Процесс построения Д-операторов не ограничен и таким образом могут быть построены производные Д-операторы неограниченной функциональности.

Учитывая эту возможность, общий случай Д-оператора, выполняющего операции ввода–вывода, определим следующим образом.

Определение 6. Общим случаем Д-операторов, осуществляющих операции ввода–вывода, будем называть Д-операторы вида $(D,D^R)\,O\,(D',D^W)$, который при пустых множествах некоторых входных и/или выходных данных трансформируется в частные случаи таких Д-операторов. Ограничения на возможности получения частных случаев задаются свойством (1). Для данных, специфицированных на входе и выходе Д-операторов, допустимо как $D^R\cap D=\varnothing$, $D^W\cap D'=\varnothing$, так и $D^R\cap D\neq\varnothing$, $D^W\cap D'\neq\varnothing$.

Теперь, с учетом определения 6, перейдем к рассмотрению композиционных схем алгоритмов.

Исходим из того, что алгоритм – это некоторый случай Д-оператора. Для определения алгоритма детализуем данные на входе и выходе Д-оператора. Заметим, что в данном случае под теоретико-множественными операциями понимаются операции над множествами носителей данных.

Определение 7. Входные и выходные данные Д-оператора $(\mathbf{D})\,\mathrm{X}\,(\mathbf{D}')$ представлены в виде следующих подмножеств:

- $\mathbf{D}=D\cup\hat{D}$, таких, что $D\cap\hat{D}=\varnothing$ и $\hat{D}\cap D'=\varnothing$;
- $\mathbf{D}=D'\cup\tilde{D}'$, таких, что $D'\cap\tilde{D}'=\varnothing$ и $\tilde{D}'\cap D=\varnothing$,

которые назовем: $\hat{D}$ – исходные, D, D' – проходные, $\tilde{D}$ – производные. При этом $D \cap D' = D = D'$. Любое из подмножеств, образующих множества $\mathbf{D}$ и $\mathbf{D}'$, может быть пустым.

Теперь определим алгоритм с учетом определения 7.

Определение 8. Произвольный автономный (не связанный с другими) алгоритм представляет собой Д-оператор $(\hat{D}, D^R)A(D^W)$, где $\hat{D}$ – исходные, D^R – вводимые данные, существующие (определенные, имеющие определенные значения) до начала функционирования алгоритма и $\hat{D} \cap D^R = \varnothing$, а D^W-данные – результат его функционирования. Множества данных $\hat{D}$ и D^R по отдельности или вместе могут быть пустыми.

Заметим, что $D^W \neq \varnothing$, так как в противном случае, в соответствии со свойством 1, $(\hat{D}, D^R)A(D^W) = Z$.

Алгоритм, в соответствии с определениями 4, 8, может быть представлен в виде КС, которую для общего случая мы и запишем

$$\begin{aligned}&(\hat{D}, D^R)A(D^W) = (\hat{D}_1, D_1^R)Q_1(D_1^W, D_1') * \\ &*(D_2, D_2^R)Q_2(D_2^W, D_2') * \ldots * (D_m, D_m^R)Q_m(D_m^W).\end{aligned}, \quad (2)$$

Все исходные, вводимые и выводимые данные «распределяются» внутри композиции Д-операторов, т.е. у

$$\hat{D} \subseteq D_1 \cup D_2 \cup \ldots \cup D_m,\ D^R = D_1^R \cup D_2^R \cup \ldots \cup D_m^R,$$
$$D^W = D_1^W \cup D_2^W \cup \ldots \cup D_m^W, \quad (3)$$

при выполнении этих условий и ограничения (1) у любого Д-оператора, входящего в КС, множества D_i, D_i^R, D_i^W могут быть пустыми. Кроме того, данные обладают следующими свойствами

$$\hat{D}_1 \subseteq \hat{D},\ D_m' = \varnothing. \quad (4)$$

Представление алгоритма в виде композиции Д-операторов будем рассматривать как первый шаг в процессе поуровневого его описания (разработки).

В результате дальнейшей детализации каждого Д-оператора, входящего в КС, на втором и всех последующих шагах будут получены семейства композиционных схем, образующие слои алгоритма. На i-м шаге слой алгоритма будет представлять собой совокупность следующих выражений

$$\begin{aligned}&{}^{i-1}(\hat{D}_1, D_1^R)Q_1(D_1^W, D_1') = {}^{i}(\hat{D}_1, D_1^R)Q_1(D_1^W, D_1') * \ldots \\ &\ldots * {}^{i}(D_{m_1}, D_{m_1}^R)Q_{m_1}(D_{m_1}^W); \\ &{}^{i-1}(\hat{D}_2, D_2^R)Q_2(D_2^W, D_2') = {}^{i}(D_{m_1+1}, D_{m_1+1}^R) \times \\ &\times Q_{m_1+1}(D_{m_1+1}^W, D_{m_1+1}') * \ldots * {}^{i}(D_{m_2}, D_{m_2}^R)Q_{m_2}(D_{m_2}^W); \\ &\ldots \\ &{}^{i-1}(\hat{D}_j, D_j^R)Q_j(D_j^W, D_j') = {}^{i}(D_{m_{j-1}+1}, D_{m_{j-1}+1}^R)Q_{m_{j-1}+1} \times \\ &\times (D_{m_{j-1}+1}^W, D_{m_{j-1}+1}') * \ldots * {}^{i}(D_{m_j}, D_{m_j}^R)Q_{m_j}(D_{m_j}^W); \\ &\ldots \\ &{}^{i-1}(\hat{D}_n, D_n^R)Q_n(D_n^W, D_n') = {}^{i}(D_{m_{n-1}+1}, D_{m_{n-1}+1}^R)Q_{m_{n-1}+1} \times \\ &\times (D_{m_{n-1}+1}^W, D_{m_{n-1}+1}') * \ldots * {}^{i}(D_{m_n}, D_{m_n}^R)Q_{m_n}(D_{m_n}^W),\end{aligned}, \quad (5)$$

для которых свойства (3), (4) записываются в виде:

$$\begin{aligned}&{}^{i-1}\hat{D}_1 \subseteq {}^{i}(D_1 \cup D_2 \cup \ldots \cup D_{m_1}), \ldots, {}^{i-1}\hat{D}_n \subseteq \\ &\subseteq {}^{i}(D_{m_{n-1}+1} \cup D_{m_{n-1}+2} \cup \ldots \cup D_{m_n}); \\ &{}^{i-1}D_1^R = {}^{i}(D_1^R \cup D_2^R \cup \ldots \cup D_{m_1}^R), \ldots, {}^{i-1}D_n^R = \\ &= {}^{i}(D_{m_{n-1}+1}^R \cup D_{m_{n-1}+2}^R \cup \ldots \cup D_{m_n}^R); \\ &{}^{i-1}D_1^W = {}^{1}(D_1^W \cup D_2^W \cup \ldots \cup D_{m_1}^W), \ldots, {}^{i-1}D_n^W = \\ &= {}^{i}(D_{m_{n-1}+1}^W \cup D_{m_{n-1}+2}^W \cup \ldots \cup D_{m_n}^W); \\ &{}^{i}\hat{D}_1 \subseteq {}^{i-1}\hat{D}_1,\ D_{m_n}' = \varnothing.\end{aligned}$$

Из изложенного видно, что алгоритм и образующие его Д-операторы, включающие операции ввода-вывода, могут быть детализованы (декомпозированы), т.е. представлены в виде совокупностей КС.

Проиллюстрируем эту возможность на примере классической для систем автоматического управления (САУ) задачи, алгоритм решения которой основан на интенсивном взаимодействии с ВУ, т.е. на использовании операций ввода-вывода.

САУ, используя некоторые исходные данные, осуществляет ввод по готовности информации с двух датчиков, обработку введенной информации и вывод управляющих воздействий на исполнительный механизм (ИМ). Пола-

гая, что датчики подключены к некоторому $УСО_1$, а ИМ – к некоторому $УСО_3$, алгоритм САУ (Д-оператор A) запишем в общем виде как $(\hat{D}, D^{R\_УСО_1})\,A\,(D^{W\_УСО_3})$.

Первый шаг детализации алгоритма представим в виде следующей КРС

$$(\hat{D}, D^{R\_УСО_1})A(D^{W\_УСО_3}) = (\varnothing)S(D^{W\_УСО}) *$$
$$*\,[1]\,\{(\hat{D}_1, D^{R\_УСО_1})\,U_1(D_1^{W\_УСО_3}) * \qquad (6)$$
$$*\,(\hat{D}_2, D^{R\_УСО_1})\,U_2(D_1^{W\_УСО_3})\},$$

где 1 – тождественно истинное логическое условие.

Из построенной КРС видно, что алгоритм САУ представляет собой две подсистемы U_1 и U_2, началу функционирования которых предшествует инициализация группы УСО, посредством Д-оператора $(\varnothing)S\,(D^{W\_УСО})$.

Каждая подсистема вводит информацию $D^{R\_УСО_1}$ с датчиков, подключенных к $УСО_1$, а результаты обработки $D_1^{W\_УСО_3}$ выводит на один ИМ, подключенный к $УСО_3$. При этом обе подсистемы функционируют в «бесконечном» цикле, что характерно для систем данного класса.

Продолжив детализацию алгоритма, получим три КРС:

- $(\varnothing)S(D^{УСО}) = (\varnothing)S_1(D^{УСО_1}) * (\varnothing)S_2(D^{УСО_3})$;
- $(\hat{D}_1, D^{R\_УСО_1})U_1(D_1^{W\_УСО_3}) = [(D_1^{R\_УСО_1})R_1(D^d) *$
  $*\,(D^d)P(\alpha^2)]\{Z\} * (D_2^{УСО_1})\,R_2(Д_1) *$
  $*\,(\hat{D}_1, Д_1)O_1(Д_1') * (Д_1')W(D_1^{W\_УСО_3})$; (7)
- $(\hat{D}_2, D^{R\_УСО_1})U_2(D_1^{W\_УСО_3}) = [(D_1^{R\_УСО_1})R_1(D^d) *$
  $*\,(D^d)P(\alpha^2)]\{Z\} * (D_3^{УСО_1})\,R_3(Д_2) *$
  $*\,(\hat{D}_2, Д_1', Д_2)O_2(Д_2') * (Д_2')W(D_1^{W\_УСО_3})$,

где $Д_1$ и $Д_2$ – данные, полученные с первого и второго датчиков, $Д_1'$ и $Д_2'$ – результаты обработки этих данных.

В первой КРС осуществляется инициализация всех УСО.

Во второй (первая подсистема) – ожидание готовности датчика до ее появления, с помощью операции p-итерации с тождественным телом цикла. Ввод данных с датчика, подключенного к первому порту $УСО_1$, и занесение их в память ($Д_1$). Обработка введенных данных, с использованием исходных данных ($\hat{D}_1$), занесение результатов обработки в память ($Д_1'$) и вывод на ИМ, подключенный к 1-му порту $УСО_3$.

В третьей (вторая подсистема) – выполняются та же последовательность действий при следующих отличиях. Данные вводятся с датчика, подключенного ко второму порту $УСО_1$, и сохраняются в памяти $Д_2'$.

Продемонстрировав на конкретном примере возможность описания алгоритмов, содержащих операции ввода-вывода, перейдем к рассмотрению информационных связей в КС.

Информационные связи в схемах алгоритмов

В [7] было введено понятие связанных Д-операторов, которое уточним в контексте данной работы.

Определение 9. Д-операторы $(D_i, D_i^R)O_i(D_i^W, D_i')$ и $(D_j, D_j^R)O_j(D_j^W, D_j')$ $(i < j)$, входящие в КС, связаны, если для них выполняется соотношение: $D_i' \cap D_j \neq \varnothing$, т.е. некоторое подмножество выходных данных Д-оператора $(D_i, D_i^R)O_i(D_i^W, D_i')$ поступает на вход Д-оператора $(D_j, D_j^R)O_j(D_j^W, D_j')$. Будем говорить, что множество данных ${}_iD_j = D_i' \cap D_j$ (${}_iD_j \subseteq D_i'$ и ${}_iD_j \subseteq D_i$) связывает операторы O_i и O_j (на что указывают используемые индексы) и эти данные назовем связывающими.

Из определений 1 и 9 следует, что динамические данные – связывающие, однако такая связь возможна только для последовательно выполняющихся Д-операторов, т.е. связанных операцией композиция.

Для того чтобы специфицировать информационные связи Д-оператора, используем следующую систему обозначений. Связывающие данные на входе j-го оператора, связывающие его с k-м, будем обозначать ${}_k\tilde{D}_j$ (где k – адрес источника, j – адрес приемника данных и $k < j$), а на выходе, связывающие его с p-м –

$_j\hat{D}_p$ (где j – адрес источника, p – адрес приемника данных и $j < p$).

Понятие информационных связей в композиционных схемах алгоритмов определим, с учетом определения 9, следующим образом.

Определение 10. Множество $S_j^{\text{л}} = {}_1\breve{D}_j, \breve{D}_j, \ldots, {}_i\breve{D}_j, \cdots, {}_{j-1}\breve{D}_j$ назовем множеством левых связей j-го оператора, а множество $S_j^{\text{п}} = {}_j\hat{D}_{j+1}, {}_j\hat{D}_{j+2}, \ldots, {}_j\hat{D}_p, \ldots, {}_j\hat{D}_n$ – множеством его правых связей в КС. Любое из подмножеств ${}_j\hat{D}_p \subseteq S_j^{\text{п}}$ может быть пустым (${}_j\hat{D}_p = \varnothing$), тогда и для ${}_j\breve{D}_p \subseteq S_j^{\text{л}}$ выполняется ${}_j\breve{D}_p = \varnothing$.

В соответствии с определениями 9, 10 для общего случая, когда каждый Д-оператор связан со всеми следующими за ним и всеми предшествующими ему Д-операторами, запишем, используя введенные обозначения, композиционную схему алгоритма (2) со связями

$$(\hat{D}, D^R)A(D'^W) = (\hat{D}_1, D_1^R)\,O_1(D_1'^W, {}_1\hat{D}_2, {}_1\hat{D}_3, \ldots, {}_1\hat{D}_j, \ldots, {}_1\hat{D}_n)*$$
$$*({}_1^{\sigma}D_2, \hat{D}_2, D_2^R)\,O_2(D_2'^W, {}_2^{\rho}D_3, \ldots, {}_2^{\rho}D_j, \ldots, {}_2^{\rho}D_n)*\ldots$$
$$\ldots*({}_1^{\sigma}D_j, \ldots, {}_i^{\sigma}D_j, \ldots, {}_{j-1}^{\sigma}D_j, \hat{D}_j, D_j^R)\,O_j(D_j'^W, {}_j^{\rho}D_{j+1}, \ldots, {}_j^{\rho}D_p, \ldots, {}_j^{\rho}D_n)*\ldots$$
$$\ldots*({}_1^{\sigma}D_n, \ldots, {}_j^{\sigma}D_n, \ldots, {}_{n-1}^{\sigma}D_n, \hat{D}_n, D_n^R)\,O_n(D_n'^W).$$

Очевидно, что связи в КС могут быть специфицированы на любом этапе детализации алгоритма. Используя эту возможность слой алгоритма (5), полученный на i-м шаге детализации, запишем в виде:

$$^{i-1}(\hat{D}_1, D_1^R)\,O_1(D_1^W, {}_1\hat{D}_2, \ldots, {}_1\hat{D}_j, \ldots, {}_1\hat{D}_n) =$$
$$^{i}(\hat{D}_1, D_1^R)O_1(D_1^W, {}_1^{\rho}D_2, \ldots, {}_1^{\rho}D_j, \ldots, {}_1^{\rho}D_{m_1}, \ldots, {}_1^{\rho}D_{m_j}, \ldots, {}_1^{\rho}D_{m_n})*\ldots$$
$$\ldots*{}^{i}({}_1^{\sigma}D_{m_1}, \ldots, {}_j^{\sigma}D_{m_1}, \ldots, {}_{m_1-1}^{\sigma}D_{m_1}, \hat{D}_{m_1}, D_{m_1}^R)\,O_{m_1}(D_{m_1}^W, {}_{m_1}^{\rho}D_{m_1+1}, \ldots, {}_{m_1}^{\rho}D_{m_n});$$
$$^{i-1}({}_1\breve{D}_2, \hat{D}_2, D_2^R)\,O_2(D_2^W, {}_2\hat{D}_3, \ldots, {}_2\hat{D}_j, \ldots, {}_2\hat{D}_n) =$$
$$= {}^{i}({}_1^{\sigma}D_{m_1+1}, \ldots, {}_j^{\sigma}D_{m_1+1}, \ldots, {}_{m_1}^{\sigma}D_{m_1+1}, \hat{D}_{m_1+1}, D_{m_1+1}^R)\,O_{m_1+1}(D_{m_1+1}^W, {}_{m_1+1}^{\rho}D_{m_1+2}, \ldots, {}_{m_1+1}^{\rho}D_{m_n})*\ldots$$
$$\ldots*{}^{i}({}_1^{\sigma}D_{m_2}, \ldots, {}_j^{\sigma}D_{m_2}, \ldots, {}_{m_2-1}^{\sigma}D_{m_2}, \hat{D}_{m_2}, D_{m_2}^R)\,O_{m_2}(D_{m_2}^W, {}_{m_2}^{\rho}D_{m_2+1}, \ldots, {}_{m_2}^{\rho}D_{m_n});$$
$$\ldots$$
$$^{i-1}({}_1\breve{D}_j, \ldots, {}_i\breve{D}_j, \ldots, {}_{j-1}\breve{D}_j, \hat{D}_j, D_j^R)\,O_j(D_j^W, {}_j\hat{D}_{j+1}, \ldots, {}_j\hat{D}_p, \ldots, {}_j\hat{D}_n) =$$
$$= {}^{i}({}_1^{\sigma}D_{m_{j-1}+1}, \ldots, {}_{m_{j-1}}^{\sigma}D_{m_{j-1}+1}, \hat{D}_{m_{j-1}+1}, D_{m_{j-1}+1}^R)\,O_{m_{j-1}+1}(D_{m_{j-1}+1}^W, {}_{m_{j-1}+1}^{\rho}D_{m_{j-1}+2}, \ldots, {}_{m_{j-1}+1}^{\rho}D_{m_n})*\ldots$$
$$\ldots*{}^{i}({}_1^{\sigma}D_{m_j}, \ldots, {}_{m_j-1}^{\sigma}D_{m_j}, \hat{D}_{m_j}, D_{m_j}^R)\,O_{m_j}(D_{m_j}^W, {}_{m_j}^{\rho}D_{m_j+1}, \ldots, {}_{m_j}^{\rho}D_{m_n});$$
$$\ldots$$
$$^{i-1}({}_1\breve{D}_n, \ldots, {}_j\breve{D}_n, \ldots, {}_{n-1}\breve{D}_n, \hat{D}_n, D_n^R)\,O_n(D_n^W) =$$
$$= {}^{i}({}_1^{\sigma}D_{m_{n-1}+1}, \ldots, {}_{m_{n-1}}^{\sigma}D_{m_{n-1}+1}, \hat{D}_{m_{n-1}+1}, D_{m_{n-1}+1}^R)\,O_{m_{n-1}+1}(D_{m_{n-1}+1}^W, {}_{m_{n-1}+1}^{\rho}D_{m_{n-1}+2}, \ldots, {}_{m_{n-1}+1}^{\rho}D_{m_n})*\ldots$$
$$\ldots*({}_1^{\sigma}D_{m_n}, \ldots, {}_{m_n-1}^{\sigma}D_{m_n}, \hat{D}_{m_n}, D_{m_n}^R)\,O_{m_n}(D_{m_n}^R).$$

В записанных КС не только специфицированы данные и связи между Д-операторами, но и все источники и приемники данных поставлены в однозначное соответствие, т.е. любому ${}_j\hat{D}_p$ соответствует ${}_j\breve{D}_p$. При этом получена возможность спецификации информационных связей между всеми Д-операторами, образующими слой алгоритма, т.е. как внутри КС, так и между ними.

Как правило, не все операторы в КС связаны друг с другом, так как в соответствии с определением 10 множества ${}_j\hat{D}_p$ (${}_j\breve{D}_p$) могут быть пустыми. Более того, при $S_j^{\text{л}} = \varnothing$ и/или $S_j^{\text{п}} = \varnothing$ Д-оператор не имеет левых и/или правых связей.

Возможность спецификации информационных связей в алгоритмах весьма существенна с учетом контроля корректности информационных связей между Д-операторами. Тем более, что такая возможность обеспечена на всех этапах разработки алгоритма.

На данные, циркулирующие в КС, с учетом связывающих, налагаются некоторые ограничения.

Утверждение. У Д-операторов, входящих в композиционную схему алгоритма, на входе, помимо вводимых, присутствуют только связывающие и исходные (у первого $(\hat{D}_1, D_1^R)\,O_1(D_1^W, D_1')$ – только исходные), а на выходе, помимо выводимых, только связывающие (у последнего $(D_n, D_n^R)O_n(D_n^W)$ – только выводимые) данные.

Доказательство следует из следующих простейших рассуждений.

Если предположить, что на выходе Д-оператора $(\hat{D}_j, {}_1\breve{D}_j, \ldots, {}_l\breve{D}_j, \ldots, {}_{j-1}\breve{D}_j, D_j^R)\,O_j(D_j', D_j'^W, {}_j\hat{D}_{j+1}, \ldots, {}_j\hat{D}_p, \ldots, {}_j\hat{D}_n)$ специфицированы данные D_j', отличные от связывающих и выводимых,

то они, очевидно, не будут использоваться на следующих шагах вычислительного процесса.

Если предположить, что на входе *j*-го Д-оператора $(D_j, \hat{D}_j, {}_1\breve{D}_j, \ldots, {}_l\breve{D}_j, \ldots, {}_{j-1}\breve{D}_j, D_j^R) O_j (D_j'^W, {}_j\hat{D}_{j+1}, \ldots, {}_j\hat{D}_p, \ldots, {}_j\hat{D}_n)$ специфицированы данные D_j, отличные от исходных, связывающих и входных, то они будут неопределенными на следующих шагах вычислительного процесса.

В обоих случаях такие данные либо избыточны, либо свидетельствуют о наличии ошибки в спецификации данных.

Утверждение доказано.

Возможность идентификации «паразитных» спецификаций, в соответствии с утверждением, еще одно положительное свойство введенных информационных связей.

Проследим применимость спецификации информационных связей для рассмотренного примера.

Первый шаг декомпозиции САУ не выявил связей между подсистемами, что легко увидеть из (6). Специфицируем информационные связи в КРС, записанных в (7). Для этого последовательно пронумеруем все производные Д-операторы:

$$(\hat{D}_1, D^{R\_\text{УСО}_1}) U_1 (D_1^{W\_\text{УСО}_3}) = {}^2[(\hat{D}_1, D_1^{R\_\text{УСО}_1}) R_1 (D^d) *$$
$$*(D^d) P_2(\alpha^2)]\{Z\} * (D_1^{\text{УСО}_1}) R_3 ({}_3\hat{\text{Д}}_4, {}_3\hat{\text{Д}}_9) *$$
$$*(\hat{D}_1, {}_3\overset{\sigma}{\text{Д}}_4) O_4 ({}_4\overset{\rho}{\text{Д}}_5) * ({}_4\overset{\sigma}{\text{Д}}_5) W_5 (D_3^{W\_\text{УСО}_3});$$
$$(\hat{D}_2, D^{R\_\text{УСО}_1}) U_2 (D_1^{W\_\text{УСО}_3}) = {}^2[(D_2^{R\_\text{УСО}_1}) R_6 (D^d) *$$
$$*(D^d) P_7(\alpha^2)]\{Z\} * (D_2^{\text{УСО}_1}) R_8 ({}_8\hat{\text{Д}}_9) *$$
$$*(\hat{D}_1, {}_3\overset{\sigma}{\text{Д}}_9) O_9 ({}_9\overset{\rho}{\text{Д}}_{10}) * ({}_9\overset{\sigma}{\text{Д}}_{10}) W_{10} (D_3^{W\_\text{УСО}_3}).$$

В обоих приведенных случаях все Д-операторы последовательно связаны, что видно, во-первых, из наличия связывающих, в частности динамических данных. Во-вторых, цикл ожидания готовности датчика, невзирая на отсутствие информационной связи, связан с Д-оператором, осуществляющим ввод данных, «семантически», так как ввод с датчика по условию задачи осуществляется по его готовности.

При этом выявлена связь между подсистемами. Множество связывающих данных ${}_3\hat{\text{Д}}_9$ (${}_3\breve{\text{Д}}_9$), полученных в первой подсистеме, используется для обработки данных во второй.

Таким образом, наряду с возможностями контроля корректности алгоритмов, открываются дополнительные возможности для их анализа.

Заключение. Очередной модификацией модели ЭВМ осуществлено дальнейшее расширение возможностей алгебры алгоритмов с данными. В результате развития алгебраического аппарата обеспечена возможность формализованного описания операций ввода-вывода в алгоритмах, записанных в виде композиционных, регулярных или композиционно-регулярных схем. Эти операции могут быть заданы для групп устройств, отдельных устройств и данных, доступных на этих устройствах, в зависимости от степени детализации описываемого алгоритма. Эта возможность особенно актуальна для управляющих и информационно-управляющих систем, интенсивно взаимодействующих с разнообразными и многочисленными внешними устройствами.

Показаны свойства схем, содержащих операции ввода-вывода, и возможность специфицировать все информационные связи как в КС алгоритмов, так и между схемами, образующими слой алгоритма. Использование таких спецификаций в качестве средства контроля корректности алгоритмов представляется весьма перспективным.

Анализ информационных связей в алгоритмах позволяет решать задачу их оптимизирующих преобразований. Отсутствие формализованного описания взаимодействия алгоритмов с ВУ ограничивало такую возможность [8]. Использование полученных в статье результатов для решения некоторых задач, оптимизирующих преобразований алгоритмов, – ближайшая цель авторов.

Кроме того, перспективные направления дальнейших исследований, использующих полученные результаты, – это задача оценки информационной сложности алгоритмов и про-

блема распараллеливания последовательных алгоритмов.

1. *Данные* в языках программирования: абстракция и типология. Сб. ст. / Под ред. В. Агафонова. – М.: Мир, 1982. – 328 с.
2. *Bastani F.B.*, *Iyengar S.S.* The effect of data structures on the logical complexity of programs // CACM, 1987, **30**. – N 3. – P. 250–259.
3. *Вирт Н.* Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
4. *Глушков В.М.*, *Цейтлин Г.Е.*, *Ющенко Е.Л.* Алгебра. Языки. Программирование. – К.: Наук. думка, 1978. – 319 с.
5. *Акуловский В.Г.* Расширенная алгебра алгоритмов. – Проблеми програмування. – 2007. – № 3 – С. 3–15.
6. *Акуловский В.Г.* Основы алгебры алгоритмов, базирующейся на данных / Матер. Сьомої міжнар. наук.-практ. конф. з програмування УкрПРОГ`2010. Київ // Проблеми програмування. – 2010. – № 2/3. – С. 89–96.
7. *Акуловский В.Г.* Некоторые аспекты формализации данных и декомпозиция Д-операторов // Там же. – 2009. – № 4 – С. 3–10.
8. *Акуловский В.Г.* Некоторые аспекты преобразования алгоритмов на основе формализации информационных связей // Кибернетика и системный анализ. – 2009. – № 6. – С. 50–54.

Поступила .03.11.2011

Тел. для справок: (044) 526-3539 (Киев)
E-mail:*dor@isofts.kiev.ua*

●

Окончание статьи М.Н. Дубовенко и др.

Заключение. Добавление агентов идентификации социальных сетей и пользователей в расширенную *MVC*-архитектуру распределенных приложений позволяет интегрировать социальные приложения в несколько социальных сетей одновременно и взаимодействовать с их пользователями без дополнительной разработки программных систем. Гибкость такой архитектуры позволит в дальнейшем расширять бизнес-логику социального приложения новыми задачами, а также адаптировать его под мобильные платформы.

Социальные приложения как направление распределенных информационных систем дают инструментальную возможность масштабным исследованиям и мониторингам в Интернете.

Управление социальными приложениями можно развивать до корпоративного уровня для решения совместных научных задач при взаимодействии двух и более организаций независимо от территориального расположения.

1. *Белов В.М.*, *Дубовенко М.Н.* К проблеме Интернет-зависимости // Кибернетика и вычислительная техника. – 2010. – **161**. – С. 53–60.
2. *http*://*vkontakte.ru*/ *developers.php#devstep*2
3. Как создавать социальные приложения для *Mail.Ru*. – *http*://*api.mail.ru*/*docs*/*guides*/*social-apps*/
4. *Apps* on Facebook.com. – *http*://*developers.facebook.com*/*docs*/*guides*/*canvas*/
5. *Соловйова К.О., Мовчан В.В*. Розробка моделі програмного засобу візуалізації мережевих структур. – *http*://*www.nbuv.gov.ua*/*portal*/*natural*/*vcpi*/*Sa*/2010_9/*statya*25_9.*pdf*
6. *Карякин О.И.*, *Кочетова Е.О.*, *Щербак С.С.* Технологии разработки распределенных приложений и их применения в социальных сетях // Нові технології. Наук. вісн. Кременчуцького ун-ту економіки, інформаційних технологій і управління. – 2008 – № 4. – С. 70–77.
7. *Дубовенко М.Н.*, *Белов В.М.* Информационная модель взаимодействия программного исследовательского комплекса с внешними средами в Интернете / Мат. науч.-тех. шк.-сем., ФМШ Жукин, 21–24 июня 2011 г. – С. 33–35.
8. *Дубовенко М.Н.*, *Белов В.М.* Концептуальный алгоритм классификации психологических проблем пользователей на основе приложений в социальных сетях // Кибернетика и вычислительная техника. – 2011. – Вып. 165. – С. 3–15.

Поступила 12.10.2011

Тел. для справок: (044) 503-9565 (Киев)
E-mail: *dep*150*@ukr.net*

●